# Bài 1. Số siêu nguyên tố Tên file: sngto.cpp

Một số tự nhiên *N* được gọi là siêu nguyên tố nếu bản thân nó là một số nguyên tố và tất cả các số thu được bằng cách xóa lần lượt các chữ số bên phải của nó đều là số nguyên tố.

Ví dụ:

Số 317 là một số siêu nguyên tố vì: 317 là 1 số nguyên tố

Xóa 1 chữ số bên phải: 31 là 1 số nguyên tố

Xóa 2 chữ số bên phải: 3 là 1 số nguyên tố

Cho số tự nhiên k (1 ≤ k ≤ 9), đưa ra số lượng số siêu nguyên tố có k chữ số.

## Input: sngto.inp:

* Dòng đầu tiên chứa số nguyên dương t – số lượng testcase
* t dòng tiếp theo, mỗi dòng chứa một số k

## Output: sngto.out:

* Gồm t dòng, mỗi dòng chứa số lượng siêu nguyên tố có k chữ số tương ứng với input

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **sngto.inp** | **sngto.out** | **Giải thích** |
| 2  1  2 | 4  9 | - các số siêu nguyên tố có 1 chữ số: 2 3 5 7  - các số siêu nguyên tố có 2 chữ số: 23 29 31 37 53 59 71 73 79 |

# Bài 2. Di chuyển tàu hỏa Tên file: TRAIN.CPP

Hình dưới là cơ cấu của một đường tàu tại một ga xe lửa:

Funnel chart

Description automatically generated

Ban đầu ở đường ray A có chứa các toa tàu được đánh số thứ tự từ 1 đến n theo thứ tự từ trái qua phải, người ta muốn chuyển các toa đó sang ray C để có được một thứ tự mới là một hoán vị của (1, 2, …, n) theo quy tắc: chỉ được đưa các toa tàu chạy theo đường ray theo hướng mũi tên, có thể dùng đường ray B để lưu tạm các toa tàu trong quá trình di chuyển.

Cho biết hoán vị cần có, hãy tìm phương án di chuyển ít nhất các toa tàu trên đường ray.

## Input: train.inp

- Dòng 1: Một số nguyên dương ***n*** (***n*** ≤ 105)

- Dòng 2: ***n*** số nguyên là hoán vị của các số từ 1 đến ***n*** – trạng thái đích của các toa tàu.

## Output: train.inp

- Nếu không có phương án thì đưa ra thông báo là NO

- Nếu có phương án thì:

+ Dòng 1 đưa ra thông báo YES

+ Các dòng tiếp theo đưa ra các bước di chuyển ít nhất các toa tàu.

Ví dụ:

|  |  |
| --- | --- |
| **train.inp** | **train.out** |
| 4  1 4 3 2 | YES  A->C  A->B  A->B  A->C  B->C  B->C |

# Bài 3. Di chuyển tàu hỏa Tên file: TRAIN1.CPP

Hình dưới là cơ cấu của một đường tàu tại một ga xe lửa:

Diagram

Description automatically generated with low confidence

Ban đầu ở đường ray A có chứa các toa tàu được đánh số thứ tự từ 1 đến n theo thứ tự từ trái qua phải, người ta muốn chuyển các toa đó sang ray C để có được một thứ tự mới là một hoán vị của (1, 2, …, n) theo quy tắc: chỉ được đưa các toa tàu chạy theo đường ray theo hướng mũi tên, có thể dùng đường ray B để lưu tạm các toa tàu trong quá trình di chuyển.

Cho biết hoán vị cần có, hãy tìm phương án di chuyển ít nhất các toa tàu trên đường ray.

## Input: train1.inp

- Dòng 1: Một số nguyên dương ***n*** (***n*** ≤ 105)

- Dòng 2: ***n*** số nguyên là hoán vị của các số từ 1 đến ***n*** – trạng thái đích của các toa tàu.

## Output: train1.inp

- Nếu không có phương án thì đưa ra thông báo là NO

- Nếu có phương án thì:

+ Dòng 1 đưa ra thông báo YES

+ Các dòng tiếp theo đưa ra các bước di chuyển ít nhất các toa tàu.

Ví dụ:

|  |  |
| --- | --- |
| **train1.inp** | **train1.out** |
| 4  1 4 2 3 | YES  A->C  A->B  A->B  A->C  B->C  B->C |

# Bài 4. Xóa xâu con Tên file: STRDEL.CPP

Zookeeper đang chơi một trò chơi. Trong trò chơi này, Zookeeper được cho một chuỗi bao gồm các chữ cái 'A' và 'B'. Anh ta có thể thực hiện xóa một chuỗi con là "AB" hoặc "BB". Khi anh ta xóa một chuỗi con như vậy, các phần còn lại của chuỗi được ghép lại với nhau.

Ví dụ, Zookeeper có thể sử dụng hai hoạt động như vậy: "AABABBA" → "AABBA" → "AAA".

Zookeeper tự hỏi chuỗi ngắn nhất mà anh ta có thể tạo ra là gì. Bạn có thể giúp anh ta tìm độ dài của chuỗi ngắn nhất?

Input: STRDEL.INP

* Dòng đầu tiên chứa số nguyên dương t (1 ≤ t ≤ 20000) – số lượng xâu cần xóa.
* t dòng tiếp theo, mỗi dòng chỉ chứa một chuỗi s gồm hai chữ cái 'A' và 'B'.
* tổng độ dài các chuỗi trong file input không vượt quá 2 \* 105 ký tự

Output: STRDEL.OUT

* Gồm t dòng, mỗi dòng đưa ra độ dài ngắn nhất của chuỗi tương ứng sau khi xóa.

Ví dụ:

|  |  |
| --- | --- |
| STRDEL.INP | STRDEL.OUT |
| 3  AAA  BABA  AABBBABBBB | 3  2  0 |

# Bài 5. NGĂN XẾP CƠ BẢN Tên file: STACK.CPP

Cho dãy *n* số nguyên *a1, a2, ..., an*. Người ta định nghĩa:

* **previous(k):** Là vị trị ***i*** nhỏ nhất thỏa mãn ***i ≤ k, aj ≥ ak*** với mọi ***j = i, i+1,..., k***
* **next(k):** Là vị trí ***i*** lớn nhất thỏa mãn ***i ≥ k, aj ≥ ak*** với mọi ***j = k, k+1,..., i***

Hãy xác định các giá trị trên với mọi *k=1,2,..., n*.

Ví dụ, nếu A=(1, 3, 2, 4, 6, 7, 3, 8, 9) thì

Previous(A) = (1, 2, 2, 4, 5, 6, 4, 8, 9)

Next(A) = (9, 2, 9, 6, 6, 6, 9, 9, 9)

## Input: stack.inp

* Dòng đầu tiên ghi *n* (n ≤ 500000)
* Tiếp theo là một số dòng ghi dãy a1, a2, ..., an. Hai số liên tiếp trên một dòng cách nhau ít nhất bởi dấu cách.

## Output: stack.out

* Gồm 2 dòng, dòng 1 ghi n số của previous và dòng 2 ghi n số của next

**Ví dụ:**

|  |  |
| --- | --- |
| Stack.inp | Stack.out |
| 9  1 3 2 4 6 7 3 8 9 | 1 2 2 4 5 6 4 8 9  9 2 9 6 6 6 9 9 9 |

# Bài 6. HÀNG ĐỢI HAI ĐẦU CƠ BẢN Tên file: DQUEUE.CPP

Cho dãy số nguyên a1, a2, ..., an. Một truy vấn (Question) trên dãy con này là một lệnh có dạng **Q(i,j)** với ý nghĩa là tìm giá trị nhỏ nhất của các phần tử trong dãy con

*ai, ai+1, ..., aj* (i≤j).

Cho *m* truy vấn Q(i1, j1), Q(i2, j2), ...., Q(im, jm) thỏa mãn:

1. i1 ≤ i2 ≤ ... ≤ im-1 ≤im
2. j1 ≤ j2 ≤ ... ≤jm-1 ≤ jm

Hãy in ra các giá trị là câu trả lời cho các truy vấn tương ứng.

## Input: dqueue.inp

* Dòng đầu tiên ghi hai số nguyên dương n, m (1≤n, m ≤100000)
* Dòng thứ hai ghi n số nguyên a1, a2, ..., an
* m dòng tiếp theo, dòng thứ k ghi hai số ik, jk thể hiện cho truy vấn thứ k (dữ liệu đảm bảo thỏa mãn điều kiện 1 và 2 ở trên)

## Output: dqueue.out

* Gồm m dòng, dòng thứ k ghi kết quả của truy vấn thứ k

Ví dụ:

|  |  |  |
| --- | --- | --- |
| **dqueue.inp** | **dqueue.inp** |  |
| 5 5  2 3 1 4 5  1 3  2 3  3 4  3 5  4 5 | 1  1  1  1  4 |  |

# Bài 7. KEYLOGGER Tên file: KEYLOGGER.CPP

John đang tìm hiểu về các công việc của một Hacker và John khá quan tâm tại sao các Hacker có thể lấy password của người dùng được. Và để tìm hiểu quá trình này, John đã muốn tự mình viết một chương trình như vậy và nó thường được gọi là KeyLogger.

Chương trình hoặt động khá đơn giản, nó ghi toàn bộ hoạt động bàn phím của người dùng. Qua trình phân tích thì sẽ tìm ra password đó.

Thật không may, một số người dùng khá tinh quái đã sử dụng thêm các phím thừa để password khó được tìm ra bằng các phím sang trái, sang phải và xóa một kí tự.

Hãy viết chương trình tìm password của người dùng để cùng so sánh với John.

## INPUT: KEYLOGGER.INP

Dữ liệu gồm một dòng duy nhất độ dài các kí tự không quá 1.000.000 thể hiện các phím mà người dùng đã ấn:

* ’-’ thể hiện xóa một kí tự đằng trước con trỏ nếu có.
* ’<’ và ’>’ thể hiện di chuyển con trỏ sang trái hoặc sang phải nếu có thể.
* Chữ cái hoặc chữ số, nó sẽ là một phần của password nếu nó không bị xóa. Nếu nó được chèn vào vị trí nào thì toàn bộ các chữ cái đằng sau sẽ dịch sang phải.

Dữ liệu cho đảm bảo password được giải mã có độ dài lớn hơn 0.

## OUTPUT: KEYLOGGER

* Ghi password của người dùng.

Ví dụ:

|  |  |
| --- | --- |
| **KEYLOGGER.INP** | **KEYLOGGER.OUT** |
| <<BP<A>>Cd- | BAPC |
| ThIsIsS3Cr3t | ThIsIsSCr3t |

# Bài 8. **Máy in Tên file: PRINT.CPP**

Máy in duy nhất trong ký túc xá sinh viên đang trải qua một khối lượng công việc rất lớn. Đôi khi có hàng trăm lệnh in trong hàng đợi máy in và bạn có thể phải đợi hàng giờ để in được tài liệu mà mình mong muốn.

Hacker General đã phát minh và triển khai một hệ thống ưu tiên đơn giản cho hàng đợi lệnh in. Hiện nay, mỗi công việc được chỉ định mức độ ưu tiên từ 1 đến 9 (với 9 là ưu tiên cao nhất, và 1 là thấp nhất), và máy in hoạt động như sau:

* + Đầu tiên lệnh ***i*** được lấy từ hàng đợi.
  + Nếu có một số lệnh in trong hàng đợi có mức độ ưu tiên cao hơn lệnh ***i***, thì lệnh ***i*** sẽ không được in và chuyển xuống dưới cuối của hàng đợi.
  + Nếu không có lệnh in nào có mức độ ưu tiên cao hơn thì thực hiện lệnh in đó (và loại bỏ nó ra khỏi hàng đợi của máy in).

Vấn đề của bạn là xác định thời gian chờ đợi để lệnh in của bạn được hoàn thành. Bạn quyết định viết một chương trình để tìm ra điều này. Chương trình sẽ được cung cấp hàng đợi hiện tại, danh sách các lệnh ưu tiên cũng như vị trí lệnh in của bạn trong hàng đợi, và sau đó chương trình của bạn sẽ tính toán mất bao lâu cho đến khi tài liệu của bạn được in, giả sử không có thêm lệnh in nào được thêm vào hàng đợi. Để đơn giản hóa vấn đề, chúng tôi giả định rằng in một tài liệu luôn mất chính xác 1 phút, việc thêm vào hàng đợi mất 1 giây, và xóa lệnh in khỏi hàng đợi diễn ra với thời gian không đáng kể.

## INPUT: PRINT.INP

* Dòng đầu tiên chứa số nguyên dương ***t*** (1 ≤ ***t*** ≤ 100) – số lượng bộ test trong input
* Mỗi bộ test có dạng như sau:
  + Dòng đầu tiên chứa hai số nguyên dương ***n*** và ***m,*** trong đó ***n*** là số lượng các lệnh in hiện thời có trong hàng đợi của máy in (1 ≤ ***n*** ≤ 100) và ***m*** là vị trí lệnh in của bạn (0 ≤ ***m*** ≤ ***n*** – 1). Vị trí đầu tiên trong hàng đợi là vị trí 0, tiếp theo là vị trí 1, …, cuối cùng là vị trí ***n*** – 1.
  + Dòng tiếp theo chứa ***n*** số nguyên từ 1 đến 9, thể hiện độ ưu tiên của các lệnh in từ vị trí 0 đến vị trí ***n*** – 1.

## OUTPUT: PRINT.OUT

* Mỗi một test trong input là thời gian (gồm 2 số cách nhau một khoảng trắng là phút và giây) chờ đợi mà lệnh in của bạn được hoàn thành.

Ví dụ:

|  |  |
| --- | --- |
| **PRINT.INP** | **PRINT.OUT** |
| 3  1 0  5  4 2  1 2 3 4  6 0  1 1 9 1 1 1 | 1 0  2 5  5 2 |

# Bài 9. **Loại bỏ những quân bài Tên file: RECARD.CPP**

Trên bàn có n quân bài được sắp xếp thành một cỗ bài, quân bài trên cùng là quân 1, quân bài dưới cùng là quân ***n***.

Hoạt động sau được thực hiện liên tục cho đến khi chỉ còn 1 quân bài:

* Loại bỏ quân bài ở trên cùng cỗ bài, sau đó quân bài mới ở trên cùng (sau khi đã loại bỏ) sẽ được chuyển xuống dưới cùng cỗ bài.

Hãy cho biết những quân bài nào đã bị loại bỏ và quân bài nào còn lại trên cỗ bài.

## INPUT: RECARD.INP

* Gồm nhiều dòng, mỗi dòng chứa một số nguyên dương n (2 ≤ n ≤ 100).

## OUTPUT: RECARD.OUT

* Với mỗi bộ test trong input tương ứng đưa ra 2 dòng:
  + Dòng 1 là danh sách các quân bài bị loại
  + Dòng 2 là quân bài còn lại trên cỗ bài.

Ví dụ:

|  |  |
| --- | --- |
| **RECARD.INP** | **RECARD.OUT** |
| 7  19  10  6 | 1 3 5 7 4 2  6  1 3 5 7 9 11 13 15 17 19 4 8 12 16 2 10 18 14  6  1 3 5 7 9 2 6 10 8  4  1 3 5 2 6  4 |

# Bài 10. GIÁ TRỊ NHỎ NHẤT Tên file: MINIMUM.CPP

Cho dãy số nguyên 𝐴 = (𝑎1, 𝑎2, … , 𝑎𝑛) và một số nguyên dương 𝑘 ≤ 𝑛. Với mỗi giá trị 𝑖 (1 ≤ 𝑖 ≤ 𝑛 − 𝑘 + 1), hãy xác định giá trị nhỏ nhất trong 𝑘 phần tử liên tiếp: 𝑎𝑖, 𝑎𝑖+1, … , 𝑎𝑖+𝑘−1.

## Input: MINIMUM.INP

* Dòng 1 chứa hai số nguyên dương 𝑛 ≤ 106, 𝑘 ≤ 𝑛 cách nhau bởi dấu cách
* Dòng 2 chứa 𝑛 số nguyên dương 𝑎1, 𝑎2, … , 𝑎𝑛 (∀𝑖: 𝑎𝑖 ≤ 106) cách nhau bởi dấu cách

## Output: MINIMUM.OUT

**Ví dụ:**

|  |  |  |
| --- | --- | --- |
| **MINIMUM.INP** | **MINIMUM.OUT** |  |
| 5 3  2 1 5 3 4 | 1  1  3 |  |

# Bài 11. Cấu trúc dữ liệu Tên file: DATA.CPP

Có một cấu trúc dữ liệu được hỗ trợ bởi hai truy vấn:

* 1 x: thêm một phần tử x vào trong cấu trúc dữ liệu
* 2 x: lấy một phần tử x trong cấu trúc dữ liệu ra

Cho trước một dãy các truy vấn với các giá trị lấy ra, bạn phải đoán xem cấu trúc dữ liệu đó thuộc loại cấu trúc dữ liệu nào. Nó có thể là stack (ngăn xếp), queue (hàng đợi), priority\_queue (hàng đợi ưu tiên) hoặc không thuộc loại nào trong số 3 loại trên.

## INPUT: DATA.INP

* Gồm nhiều test, mỗi test có dòng 1 chứa số nguyên ***n*** (1 ≤ ***n*** ≤ 1000) – số lượng các truy vấn.
* ***n*** dòng tiếp theo, mỗi dòng chứa 2 số nguyên ***k*** và ***x*** – trong đó ***k*** = 1 hoặc ***k*** = 2 thể hiện truy vấn loại 1 hoặc loại 2, x là số nguyên dương không vượt quá 100.

## OUTPUT: DATA.OUT

* Mỗi test đưa ra câu trả lời:

|  |  |
| --- | --- |
| stack | Nếu nó là stack |
| queue | Nếu nó là hàng đợi |
| priority\_queue | Nếu nó là hàng đợi ưu tiên |
| impossible | Nếu nó không là ngăn xếp, không là hàng đợi hoặc không là hàng đợi ưu tiên |
| no sure | Nếu nó có thể cùng lúc là 2 trong 3 kiểu dữ liệu trên. |

Ví dụ:

|  |  |
| --- | --- |
| **DATA.INP** | **DATA.OUT** |
| 6  1 1  1 2  1 3  2 1  2 2  2 3  6  1 1  1 2  1 3  2 3  2 2  2 1  2  1 1  2 2  4  1 2  1 1  2 1  2 2  7  1 2  1 5  1 1  1 3  2 5  1 4  2 4 | queue  no sure  impossible  stack  priority\_queue |